

Complexity of checking whether two automata are synchronized by the same language

Marina Maslennikova*

Ural Federal University, Ekaterinburg, Russia
maslennikova.marina@gmail.com

Abstract. A deterministic finite automaton \mathcal{A} is said to be *synchronizing* if it has a *reset* word, i.e. a word that brings all states of the automaton \mathcal{A} to a particular one. We prove that it is a **PSPACE**-complete problem to check whether the language of reset words for a given automaton coincides with the language of reset words for some particular automaton.

Keywords: ideal language, synchronizing automaton, reset word, reset complexity, **PSPACE**-completeness.

Introduction

Let $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ be a *deterministic finite automaton* (DFA), where Q is the *state set*, Σ stands for the *input alphabet*, and $\delta : Q \times \Sigma \rightarrow Q$ is the totally defined *transition function* defining the action of the letters in Σ on Q . The function δ is extended uniquely to a function $Q \times \Sigma^* \rightarrow Q$, where Σ^* stands for the free monoid over Σ . The latter function is still denoted by δ . In the theory of formal languages the definition of a DFA usually includes the *initial state* $q_0 \in Q$ and the set $F \subseteq Q$ of *terminal states*. We will use this definition when dealing with automata as devices for recognizing languages. A language $L \subseteq \Sigma^*$ is *recognized* (or *accepted*) by an automaton $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ if $L = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. We denote by $L[\mathcal{A}]$ the language accepted by the automaton \mathcal{A} .

A DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action leaves the automaton in one particular state no matter at which state in Q it is applied: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$. Any word w with this property is said to be *reset* for the DFA \mathcal{A} . For the last 50 years synchronizing automata received a great deal of attention. In 1964 Černý conjectured that every synchronizing automaton with n states possesses a reset word of length at most $(n-1)^2$. Despite intensive efforts of researchers this conjecture still remains open. For a brief introduction to the theory of synchronizing automata we refer the reader to the recent surveys [11, 13].

In the present paper we focus on some complexity aspects of the theory of synchronizing automata. We denote by $\text{Syn}(\mathcal{A})$ the language of reset words for a

* The author acknowledges support from the Presidential Programm for young researchers, grant MK-3160.2014.1.

given automaton \mathcal{A} . It is well known that $\text{Syn}(\mathcal{A})$ is regular [13]. Furthermore, it is an *ideal* in Σ^* , i.e. $\text{Syn}(\mathcal{A}) = \Sigma^* \text{Syn}(\mathcal{A}) \Sigma^*$. On the other hand, every regular ideal language L serves as the language of reset words for some automaton. For instance, the minimal automaton recognizing L is synchronized exactly by L [7]. Thus synchronizing automata can be considered as a special representation of an ideal language. Effectiveness of such a representation was addressed in [7]. The *reset complexity* $rc(L)$ of an ideal language L is the minimal possible number of states in a synchronizing automaton \mathcal{A} such that $\text{Syn}(\mathcal{A}) = L$. Every such automaton \mathcal{A} is called a *minimal synchronizing automaton* (for brevity, MSA). Let $sc(L)$ be the number of states in the minimal automaton recognizing L . For every ideal language L we have $rc(L) \leq sc(L)$ [7]. Moreover, there are languages L_n for every $n \geq 3$ such that $rc(L_n) = n$ and $sc(L_n) = 2^n - n$ [7]. Thus the representation of an ideal language by means of a synchronizing automaton can be exponentially more succinct than the “traditional” representation via the minimal automaton. However, no reasonable algorithm is known for computing an MSA of a given language. One of the obstacles is that an MSA is not uniquely defined. For instance, there is a language with at least two different MSAs [7].

Let L be an ideal regular language over Σ with $rc(L) = n$. The latter equality means that there exists some n -state DFA \mathcal{B} such that $\text{Syn}(\mathcal{B}) = L$, and \mathcal{B} is an MSA for L . Now it is quite natural to ask the following question: how hard is it to verify the condition $\text{Syn}(\mathcal{B}) = L$? It is well known that the equality of the languages accepted by two given DFAs can be checked in polynomial of the size of automata time. However, the problem of checking the equality of the languages of reset words of two synchronizing DFAs turns out to be hard. Moreover, it is hard to check whether one particular ideal language serves as the language of reset words for a given synchronizing automaton. We state formally the SYN-EQUALITY problem:

- Input*: synchronizing automata \mathcal{A} and \mathcal{B} .
- Question*: is $\text{Syn}(\mathcal{A}) = \text{Syn}(\mathcal{B})$?

We prove that SYN-EQUALITY is a **PSPACE**-complete problem. Actually, we prove a stronger result, that it is a **PSPACE**-complete problem to check whether the language $\text{Syn}(\mathcal{A})$ for a given automaton \mathcal{A} coincides with the language $\text{Syn}(\mathcal{B})$ for some particular automaton \mathcal{B} . Also it is interesting to understand how hard is it to verify a strict inclusion $\text{Syn}(\mathcal{A}) \subsetneq \text{Syn}(\mathcal{B})$. We prove that it is not easier than to check the precise equality of the languages $\text{Syn}(\mathcal{A})$ and $\text{Syn}(\mathcal{B})$. So the problem of constructing an MSA for a given ideal language is unlikely to be an easy task. Also we obtain that the problem of checking the inequality $rc(L) \leq \ell$, for a given positive integer number ℓ , is **PSPACE**-complete. Here an ideal language L is presented by a DFA, for which L serves as the language of reset words. Actually, we prove that the problem of checking the equalities $rc(L) = 1$ or $rc(L) = 3$ is trivial, however it is a **PSPACE**-complete problem to verify whether $rc(L) = 3$.

The paper is organized as follows. In Section 1 we introduce some definitions and state formally the considered problems. In Section 2 we prove main results about **PSPACE**-completeness of the problem SYN-EQUALITY and **PSPACE**-

completeness of the problem of checking whether the reset complexity of a given ideal language is not greater than ℓ .

1 Preliminaries

A standard tool for finding the language of synchronizing words of a given DFA $\mathcal{A} = \langle Q, \delta, \Sigma \rangle$ is the *power automaton* $\mathcal{P}(\mathcal{A})$. Its state set is the set \mathcal{Q} of all nonempty subsets of Q , and the transition function is defined as a natural extension of δ on the set $\mathcal{Q} \times \Sigma$ (the resulting function is also denoted by δ), namely, $\delta(S, a) = \{\delta(s, a) \mid s \in S\}$ for $S \subseteq Q$ and $a \in \Sigma$. If we take the set Q as the initial state and singletons as final states in $\mathcal{P}(\mathcal{A})$, then we obtain an automaton recognizing $\text{Syn}(\mathcal{A})$. It is easy to see that if all the singletons identified to get a unique sink state s (i.e. s is fixed by all letters in Σ), the resulting automaton still recognizes $\text{Syn}(\mathcal{A})$. Throughout the paper the term *power automaton* and the notation $\mathcal{P}(\mathcal{A})$ will refer to this modified version.

One may notice now that the problem SYN-EQUALITY can be solved by the following naive algorithm. Indeed, we construct the power automata $\mathcal{P}(\mathcal{A})$ and $\mathcal{P}(\mathcal{B})$ for DFAs \mathcal{A} and \mathcal{B} . Now it remains to verify that automata $\mathcal{P}(\mathcal{B})$ and $\mathcal{P}(\mathcal{A})$ accept the same language. However, the automaton $\mathcal{P}(\mathcal{A})$ has $2^n - n$ states, where n is the number of states in the DFA \mathcal{A} . So we cannot afford to construct directly the corresponding power automata. Now we state formally the SYN-INCLUSION problem. It will be shown that SYN-INCLUSION is in **PSPACE**.

SYN-INCLUSION

–*Input*: synchronizing automata \mathcal{A} and \mathcal{B} .

–*Question*: is $\text{Syn}(\mathcal{A}) \subseteq \text{Syn}(\mathcal{B})$?

Since SYN-INCLUSION belongs to the class **PSPACE**, we obtain that SYN-EQUALITY is in **PSPACE** as well. Further we prove that the SYN-EQUALITY problem is complete for the class **PSPACE**. Now it is interesting to consider the SYN-STRICT-INCLUSION problem:

–*Input*: synchronizing automata \mathcal{A} and \mathcal{B} .

–*Question*: is $\text{Syn}(\mathcal{A}) \subsetneq \text{Syn}(\mathcal{B})$?

It will be shown that SYN-STRICT-INCLUSION is a **PSPACE**-complete problem.

Recall that the word $u \in \Sigma^*$ is a *prefix* (*suffix* or *factor*, respectively) of the word w if $w = us$ ($w = tu$ or $w = tus$, respectively) for some $t, s \in \Sigma^*$. A reset word w for a DFA \mathcal{A} is called *minimal* if none of its proper prefixes nor suffixes is reset. We will denote by $w[i]$ the i^{th} letter of w and by $|w|$ the length of the word w . In what follows the word $w[i]w[i+1]...w[j]$, for $i < j$, will be denoted by $w[i..j]$.

2 PSPACE-completeness

Theorem 1. *SYN-INCLUSION is in PSPACE.*

Proof. Savitch's theorem states that **PSPACE**=**NPSPACE** [12]. Therefore, it is enough to prove that SYN-INCLUSION belongs to **NPSPACE**, i.e. it suffices to solve the problem by a non-deterministic algorithm within polynomial space. Let $\mathcal{A} = \langle Q_1, \Sigma, \delta_1 \rangle$ and $\mathcal{B} = \langle Q_2, \Sigma, \delta_2 \rangle$ be synchronizing automata over Σ . We have to prove that the language $\text{Syn}(\mathcal{B})$ contains the language $\text{Syn}(\mathcal{A})$, or equivalently the following equality takes place: $\text{Syn}(\mathcal{A}) \cap \text{Syn}(\mathcal{B})^c = \emptyset$, where $\text{Syn}(\mathcal{B})^c$ is the complement language of $\text{Syn}(\mathcal{B})$. An obstacle is that we cannot afford to construct the automaton recognizing the language $\text{Syn}(\mathcal{A}) \cap \text{Syn}(\mathcal{B})^c$ directly. Instead we provide an algorithm that guesses a word w which is reset for \mathcal{A} and is not reset for \mathcal{B} . Let us notice that w may turn out to be exponentially long in $|Q_1| + |Q_2|$. Hence even if our algorithm correctly guesses w , it would not have enough space to store its guess. Thus the algorithm should guess w letter by letter.

Let $Q_1 = \{q_1, \dots, q_n\}$ and $Q_2 = \{p_1, \dots, p_m\}$. The algorithm guesses the first letter $w[1]$ of w , applies $w[1]$ at every state in Q_1 and Q_2 and stores two lists of images, namely, $\{\delta_1(q_1, w[1]), \dots, \delta_1(q_n, w[1])\}$ and $\{\delta_2(p_1, w[1]), \dots, \delta_2(p_m, w[1])\}$. These lists clearly require only $O(n + m)$ space. Further the algorithm guesses the second letter $w[2]$ and updates the lists of images re-using the space, and so on. Note that $\delta_1(q_i, w[1..k]) = \delta_1(\delta_1(q_i, w[1..k-1]), w[k])$, where $k \geq 2$ and $i \in \{1, \dots, n\}$. So we do not need to store the whole word w in order to build the sets $\delta_1(Q_1, w)$ and $\delta_2(Q_2, w)$. At the end of the guessing steps the algorithm gets two lists $\{\delta_1(q_1, w), \dots, \delta_1(q_n, w)\}$ and $\{\delta_2(p_1, w), \dots, \delta_2(p_m, w)\}$. It remains to check the following conditions:

- all the states in the first list coincide with some particular state from Q_1 ;
- there are at least two different states in the second list.

The latter checking does not require any additional space. Thus the problem SYN-INCLUSION is in **PSPACE**. \square

Since $\text{Syn}(\mathcal{A}) = \text{Syn}(\mathcal{B})$ if and only if $\text{Syn}(\mathcal{A}) \subseteq \text{Syn}(\mathcal{B})$ and $\text{Syn}(\mathcal{B}) \subseteq \text{Syn}(\mathcal{A})$, we obtain the following corollary.

Corollary 1. *SYN-EQUALITY is in **PSPACE**.*

To prove that SYN-EQUALITY is a **PSPACE**-complete problem we reduce the following well-known **PSPACE**-complete problem to the complement of SYN-EQUALITY. This problem deals with checking emptiness of the intersection of languages accepted by DFAs from a given collection [5].

FINITE AUTOMATA INTERSECTION

–*Input:* given n DFAs $M_i = \langle Q_i, \Sigma, \delta_i, q_i, F_i \rangle$, for $i = 1, \dots, n$.

–*Question:* is $\bigcap_i L[M_i] \neq \emptyset$?

Given an instance of FINITE AUTOMATA INTERSECTION, we can suppose without loss of generality that each initial state q_i has no incoming edges and $q_i \notin F_i$. Indeed, excluding the case for which the empty word ε is in $L[M_i]$ we can always build a DFA $M'_i = \langle Q'_i, \Sigma, \delta'_i, q'_i, F_i \rangle$, which recognizes the same language of M_i , such that the initial state q'_i has no incoming edges. This can be easily achieved by adding a new initial state q'_i to the state set Q_i and defining the transition function δ'_i by the rule: $\delta'_i(q'_i, a) = \delta_i(q_i, a)$ for all $a \in \Sigma$ and

$\delta'_i(q, a) = \delta_i(q, a)$ for all $a \in \Sigma$, $q \in Q_i$. Furthermore, we may assume that the sets Q_i , for $i = 1, \dots, n$, are pairwise disjoint.

To build an instance of SYN-EQUALITY from DFAs M_i , $i = 1, \dots, n$, we construct a DFA $\mathcal{A} = \langle Q, \Delta, \varphi, \rangle$ with $Q = \bigcup_{i=1}^n Q_i \cup \{s, h\}$, where s and h are new states not belonging to any Q_i . We add three new letters to the alphabet Σ and get in this way $\Delta = \Sigma \cup \{x, y, z\}$. The transition function φ of the DFA \mathcal{A} is defined by the following rules:

$$\begin{array}{ll} \varphi(q, a) = \delta_i(q, a) & \text{for all } i = 1, \dots, n, \text{ for all } q \in Q_i \text{ and } a \in \Sigma; \\ \varphi(q, x) = q_i & \text{for all } i = 1, \dots, n, \text{ for all } q \in Q_i; \\ \varphi(q, z) = s & \text{for all } i = 1, \dots, n, \text{ for all } q \in F_i; \\ \varphi(q, z) = h & \text{for all } i = 1, \dots, n, \text{ for all } q \in (Q_i \setminus F_i); \\ \varphi(q, y) = s & \text{for all } i = 1, \dots, n, \text{ for all } q \in Q_i; \\ \varphi(h, a) = s & \text{for all } a \in \Delta; \\ \varphi(s, a) = s & \text{for all } a \in \Delta. \end{array}$$

The resulting automaton \mathcal{A} is shown schematically in the Fig. 1. The action of letters from Σ on the states $p \in Q_i$ is not shown. Denote by G_i the set $Q_i \setminus (F_i \cup \{q_i\})$. All the states from the set G_i are shown as the node labeled by G_i . All the states from the set F_i are shown as the node labeled by F_i .

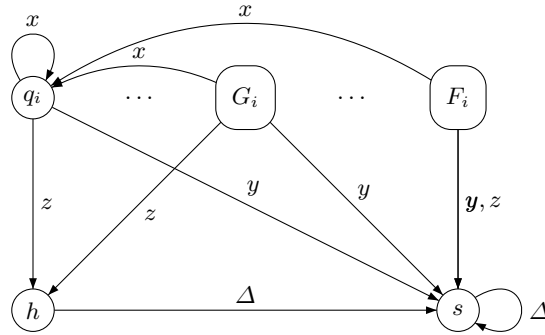


Fig. 1. Automaton \mathcal{A}

It can be easily seen that by the definition of the transition function φ we get $\varphi(Q, w) \cap Q_i \neq \emptyset$ if and only if $w \in (\Sigma \cup \{x\})^*$. From this observation and the definition of φ we obtain the following lemma.

Lemma 1. *For any $w \in \Delta^*$ we have $\varphi(Q, w) \cap Q_i \neq \emptyset$ for all $i = 1, \dots, n$ if and only if there is some $j \in \{1, \dots, n\}$ such that $\varphi(Q, w) \cap Q_j \neq \emptyset$.*

Consider the languages L_1 and L_2 :

$$\begin{aligned} L_1 &= (\Sigma \cup \{x\})^* y \Delta^*; \\ L_2 &= (\Sigma \cup \{x\})^* z \Delta^+. \end{aligned}$$

Consider the language $I = L_1 \cup L_2$.

Lemma 2. $\bigcap_{i=1}^n L[M_i] = \emptyset$ if and only if $\text{Syn}(\mathcal{A}) = I$.

Proof. Let $\bigcap_{i=1}^n L[M_i] = \emptyset$. We take a word $w \in \text{Syn}(\mathcal{A})$. Since s is a sink state in \mathcal{A} , we have $\varphi(Q, w) = \{s\}$. If $w \in (\Sigma \cup \{x\})^+$, then $\varphi(Q, w) \cap Q_i \neq \emptyset$, which is a contradiction. Therefore, w contains some factor belonging to $\{y, z\}^+$. Thus we can factorize w as $w = uav$, where u is a maximal prefix of w belonging to $(\Sigma \cup \{x\})^*$, $a \in \{y, z\}$ and $v \in \Delta^*$.

Case 1: $a = y$.

Since y maps all the states of \mathcal{A} to a sink state s , we have that w is a reset word and $w \in L_1$.

Case 2: $a = z$.

2.1. Let $u \in \Sigma^*$, i.e. u does not contain a factor belonging to $\{x\}^+$. By lemma 1, $\varphi(Q, u) \cap Q_i \neq \emptyset$ for all $i = 1, \dots, n$. Note that $u \notin \bigcap_i L_i$, that is $u \notin L_j$ for some j . It means that $\varphi(q_j, u) \in Q_j \setminus F_j$. Hence $h \in \varphi(Q, uz)$. More precisely, $\varphi(Q, uz) = \{h, s\}$. It remains to apply a letter from Δ in order to map h to s . So we have $w \in L_2$.

2.2. Let u contain a factor belonging to $\{x\}^+$. Obviously, we may factorize u as $u = u'xt$, where t is the maximal suffix of u belonging to Σ^* . By lemma 1, $\varphi(Q, u') \cap Q_i \neq \emptyset$ for all $i = 1, \dots, n$. Thus $q_i \in \varphi(Q, u'x)$ for all $i = 1, \dots, n$. By the argument from the previous case, $\varphi(q_j, u'xt) \in Q_j \setminus F_j$ for some index j . Thus $\varphi(Q, u'xtz) = \{h, s\}$ and $w \in L_2$.

So we obtain that if $\bigcap_{i=1}^n L[M_i] = \emptyset$, then $\text{Syn}(\mathcal{A}) \subseteq I$. The opposite inclusion $I \subseteq \text{Syn}(\mathcal{A})$ follows easily from the arguments above. Assume now that the equality $\text{Syn}(\mathcal{A}) = I$ takes place. Arguing by contradiction, assume that $\bigcap_{i=1}^n L[M_i] \neq \emptyset$, thus there is some $w' \in \bigcap_{i=1}^n L[M_i]$. By the definition of φ we get that the word $w = xw'z$ is a reset word for \mathcal{A} . However, $w \notin I$. So we came to a contradiction. \square

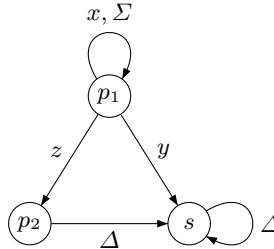


Fig. 2. Automaton \mathcal{B}

Now we build a 3-state automaton $\mathcal{B} = \langle P, \Delta, \tau \rangle$ (see Fig. 2). Its state set is $P = \{p_1, p_2, s\}$, where s is the unique sink state. Further we verify that I serves as the language of reset words for \mathcal{B} . Moreover, \mathcal{B} is an MSA for I .

Lemma 3. $\text{Syn}(\mathcal{B}) = I$.

Proof. It is clear that $I \subseteq \text{Syn}(\mathcal{B})$. Let $w \in \text{Syn}(\mathcal{B})$. Obviously, $w \notin (\Sigma \cup \{x\})^*$. Thus we may factorize w as $w = uav$, where $(\Sigma \cup \{x\})^*$, $a \in \{y, z\}$ and $v \in \Delta^*$. If $a = y$ then $w \in L_1$. If $a = z$ then $\tau(Q, uz) = \{p_2, s\}$. Since w is a reset word for \mathcal{B} we obtain that $w \in L_2$. So we have the inclusion $\text{Syn}(\mathcal{B}) \subseteq I$. \square

For each instance of FINITE AUTOMATA INTERSECTION one may construct the corresponding automaton \mathcal{A} and the DFA \mathcal{B} . It is easy to check that I does not serve as the language of reset words for a synchronizing automaton of size at most two over the same alphabet Δ . So \mathcal{B} is an MSA for I and $rc(\text{Syn}(\mathcal{B})) = 3$. Furthermore, \mathcal{B} is a finitely generated synchronizing automaton, that is its language of reset words $\text{Syn}(\mathcal{B})$ can be represented as $\text{Syn}(\mathcal{B}) = \Delta^* U \Delta^*$ for some finite set of words U . Namely, $U = y \cup z \Delta$. Finitely generated synchronizing automata and its languages of reset words were studied in [4, 9, 10]. In particular, it was shown in [9] that recognizing finitely generated synchronizing automata is a **PSPACE**-complete problem. Finally, by lemmas 2 and 3, we have the following claim.

Lemma 4. $\bigcap_{i=1}^n L[M_i] = \emptyset$ if and only if $\text{Syn}(\mathcal{A}) = \text{Syn}(\mathcal{B})$.

Now we are in position to prove the main result.

Theorem 2. *SYN-EQUALITY is **PSPACE**-complete.*

Proof. By Corollary 1, SYN-EQUALITY is in **PSPACE**. Since the construction of automata \mathcal{A} and \mathcal{B} can be performed in polynomial time from the automata M_i ($i = 1, \dots, n$), by Lemmas 2, 3 and 4, we can reduce FINITE AUTOMATA INTERSECTION to co-SYN-EQUALITY. \square

Theorem 3. *SYN-STRICT-INCLUSION is **PSPACE**-complete.*

Proof. By theorem 1, SYN-STRICT-INCLUSION is in **PSPACE**. We show that SYN-STRICT-INCLUSION is reduced to the SYN-EQUALITY problem and vice versa. Let $\mathcal{A} = \langle Q_1, \Sigma, \delta_1 \rangle$ and $\mathcal{B} = \langle Q_2, \Sigma, \delta_2 \rangle$ be synchronizing DFAs with $\text{Syn}(\mathcal{A}) = L$ and $\text{Syn}(\mathcal{B}) = L'$. Define the automaton $\mathcal{A} \times \mathcal{B} = \langle Q, \Sigma, \delta \rangle$ in the following way:

$$Q = Q_1 \times Q_2;$$

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

for all $(q_1, q_2) \in Q$ and $a \in \Sigma$. Clearly, $\text{Syn}(\mathcal{A} \times \mathcal{B}) = L \cap L'$. Thus $L \subsetneq L'$ if and only if $L = L \cap L'$ and $L \neq L'$. On the other hand, $L \neq L'$ if and only if either $L \cap L' \subsetneq L$ or $L \cap L' \subsetneq L'$. \square

Let us note that we build synchronizing automata \mathcal{A} and \mathcal{B} over at least 5-letter alphabet to obtain an instance of SYN-EQUALITY. What about alphabets of size less than five? It can be easily seen that, for automata over a unary alphabet, SYN-EQUALITY can be solved in polynomial time. Indeed, if $\mathcal{A} = \langle Q_1, \{a\}, \delta_1 \rangle$ is a synchronizing DFA, then $\text{Syn}(\mathcal{A}) = a^*a^k$, where k is the length of the shortest reset word for \mathcal{A} . Furthermore, it is easy to check that $k < |Q_1|$. Analogously, the language of reset words for a DFA $\mathcal{B} = \langle Q_2, \{a\}, \delta_2 \rangle$ is $\text{Syn}(\mathcal{B}) = a^*a^m$, where $m < |Q_2|$. Finally, positive integer numbers k and m can be found in polynomial time. So it is interesting to consider automata over alphabets of size at least two.

We have reduced the problem FINITE AUTOMATA INTERSECTION to the problem SYN-EQUALITY. By construction of DFAs \mathcal{A} and \mathcal{B} , we have $\Delta = \{y, z, a, b, x\}$. We build DFAs $\mathcal{C} = \langle C, \{\mu, \lambda\}, \varphi_2 \rangle$ and $\mathcal{D} = \langle D, \{\mu, \lambda\}, \tau_2 \rangle$ with unique sink states ζ_1 and ζ_2 respectively. It will be shown that $\text{Syn}(\mathcal{A}) = \text{Syn}(\mathcal{B})$ if and only if $\text{Syn}(\mathcal{C}) = \text{Syn}(\mathcal{D})$. A standard technique is applied here and also was used in [1, 6, 7]. Namely, we define morphisms $h : \{\lambda, \mu\}^* \lambda \rightarrow \Delta^*$ and $\bar{h} : \Delta^* \rightarrow \{\lambda, \mu\}^* \lambda$ preserving the property of being a reset word for the corresponding automaton. Let $d_1 = y, d_2 = z, d_3 = a, d_4 = b$ and $d_5 = x$. We put $h(\mu^k \lambda) = d_{k+1}$ for $k = 0, \dots, 4$ and $h(\mu^k \lambda) = d_5 = x$ for $k \geq 5$. Every word from the set $\{\lambda, \mu\}^* \lambda$ can be uniquely factorized by words $\mu^k \lambda, k = 0, 1, 2, \dots$, thus the mapping h is totally defined. We also consider the morphism $\bar{h} : \Delta^* \rightarrow \{\lambda, \mu\}^* \lambda$ defined by the rule $\bar{h}(d_k) = \mu^{k-1} \lambda$. Note that for every word $u \in \Delta^*$ we have $h(\bar{h}(u)) = u$.

Now we take the constructed above DFA $\mathcal{B} = \langle P, \Delta, \tau \rangle$ with the state set $P = \{p_1, p_2, s\}$. We build the DFA $\mathcal{D} = \langle D, \{\mu, \lambda\}, \tau_2 \rangle$ with a unique sink state ζ_2 .

We associate each state p_i of the automaton \mathcal{B} with the 5-element set of states $P_i = \{p_{i,1}, \dots, p_{i,5}\}$ of the automaton \mathcal{D} . Namely, the states $p_{i,2}, p_{i,3}, p_{i,4}, p_{i,5}$ are copies of the state p_i associated with $p_{i,1}$. The action of the letter μ is defined in the following way: $\tau_2(p_{i,k}, \mu) = p_{i,k+1}$ for $k \leq 4$, and $\tau_2(p_{i,5}, \mu) = p_{i,5}$. We put $D = P_1 \cup P_2 \cup \{\zeta_2\}$, where ζ_2 is a unique sink state. The action of the letter λ is defined by the rules:

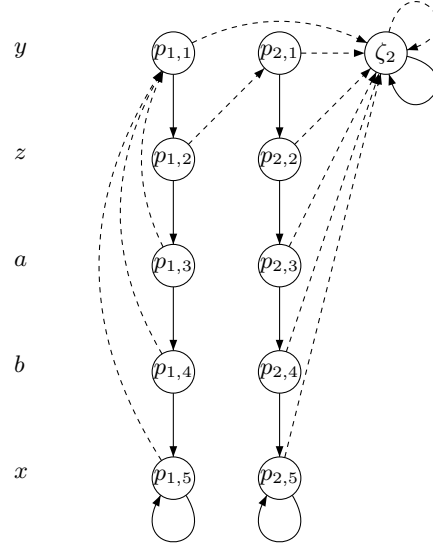
- if $\tau(p_i, d_k) = s$, then $\tau_2(p_{i,k}, \lambda) = \zeta_2$;
- if $\tau(p_i, d_k) = p_j$, then $\tau_2(p_{i,k}, \lambda) = p_{j,1}$.

The latter rule means that if there is the transition from p_i to p_j labeled by the letter d_k , then there is the transition from $p_{i,1}$ to $p_{j,1}$ labeled by the word $\mu^{k-1} \lambda$.

P_i is called the i -th column of the set D . For each $k = 1, \dots, 5$, one may take the set $R_k = \{p_{1,k}, p_{2,k}\}$. The set R_k is called the k -th row of the set D .

The DFA \mathcal{C} is constructed in analogous way. Finally, note that the resulting automata \mathcal{C} and \mathcal{D} have $O(5|Q_1|)$ and $O(5|Q_2|)$ states respectively, where $|Q_1|$ and $|Q_2|$ are the cardinalities of the state sets of \mathcal{A} and \mathcal{B} respectively. Figure 3 illustrates the automaton \mathcal{D} . The action of the letter μ is shown in solid lines, the action of the letter λ is shown in dotted lines.

Lemma 5. $\text{Syn}(\mathcal{A}) = \text{Syn}(\mathcal{B})$ if and only if $\text{Syn}(\mathcal{C}) = \text{Syn}(\mathcal{D})$.

Fig. 3. Automaton \mathcal{D}

Proof. It is convenient to organize the constructed DFA \mathcal{D} as a table. The k -th row contains copies of all states corresponding to the k -th letter from Δ . The i -th column contains the state $p_{i,1}$ corresponding to the state p_i and its copies $p_{i,2}, \dots, p_{i,5}$. Each state from the i -th column maps under the action of μ to a state from the same column. The k -th row R_k maps under the action of μ to the $k+1$ -st row R_{k+1} ($k \leq 4$). The 5-th row is fixed by μ , that is $\tau_2(R_5, \mu) = R_5$. The state set D maps under the action of λ to a subset of R_1 . The DFA \mathcal{C} possesses such properties as well.

It can be easily checked that the word $u \in \Delta^*$ is reset for the DFA \mathcal{B} if and only if $\bar{h}(u)$ is reset for \mathcal{D} . An analogous property takes place for DFAs \mathcal{A} and \mathcal{C} .

Assume that $\text{Syn}(\mathcal{A}) \neq \text{Syn}(\mathcal{B})$. From the proof of lemma 2 it follows that the word $w = xw'z$ with $w' \in \bigcap_i L[M_i]$ is reset for \mathcal{A} and it is not reset for \mathcal{B} . Thus $\bar{h}(w) \in \text{Syn}(\mathcal{C})$ and $\bar{h}(w) \notin \text{Syn}(\mathcal{D})$. So $\text{Syn}(\mathcal{C}) \neq \text{Syn}(\mathcal{D})$.

Assume now that $\text{Syn}(\mathcal{A}) = \text{Syn}(\mathcal{B})$. We show that every minimal reset word of \mathcal{C} is reset for \mathcal{D} and every minimal reset word of \mathcal{D} is reset for \mathcal{A} . Let u be a minimal reset word of \mathcal{C} . Any word $u \in \{\mu\}^*$ is not in $\text{Syn}(\mathcal{C})$, since μ brings each column to its subset. Thus we have $u \in \{\lambda, \mu\}^* \setminus \{\mu\}^*$. The automaton \mathcal{C} possesses a unique sink state ζ_1 . Hence \mathcal{C} is synchronized to ζ_1 . Furthermore, all the transitions leading to ζ_1 are labeled by λ , and ζ_1 is fixed by μ and λ . Thus if u does not end with λ then it is not a minimal reset word. We have $u \in \{\lambda, \mu\}^* \lambda$. Consider the word $w = h(u)$. Since $\bar{h}(w) = u$, we have that w is a reset word

for \mathcal{A} and \mathcal{B} . Hence $u \in \text{Syn}(\mathcal{D})$. So we obtain that $\text{Syn}(\mathcal{C}) \subseteq \text{Syn}(\mathcal{D})$. The opposite inclusion is verified analogously. \square

Lemma 5 gives the desired result on **PSPACE**-completeness of the problem SYN-EQUALITY restricted to a binary alphabet case.

Proposition 1. *Let ℓ be a positive integer number, L an ideal language, and \mathcal{A} a synchronizing DFA for which L serves as the language of reset words. The problem of checking the inequality $rc(L) \leq \ell$ is in **PSPACE**.*

Proof. If ℓ is greater or equal to the size of \mathcal{A} , then the answer is “yes” and there is nothing to prove. Let ℓ be less than the size of \mathcal{A} . One may non-deterministically guess a DFA \mathcal{B} with at most ℓ states and check the equality $\text{Syn}(\mathcal{B}) = \text{Syn}(\mathcal{A})$ within polynomial space. \square

Lemma 6. *Let L be an ideal language and \mathcal{A} some automaton with $\text{Syn}(\mathcal{A}) = L$. The equalities $rc(L) = 1$ and $rc(L) = 2$ can be checked in polynomial of the size of \mathcal{A} time.*

Proof. Let $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$. Denote by k the size of the alphabet Σ . It is easy to see that $rc(L) = 1$ if and only if $L = \Sigma^*$, so it is required that $n = 1$.

Let us notice that some 2-state automaton $\mathcal{B} = \langle P, \Sigma, \delta \rangle$ is synchronizing if and only if some letter brings the automaton to a singleton and each letter $a \in \Sigma$ either maps the state set P to a singleton or acts as a permutation on P . So we find the set $\Gamma = \{a \mid a \in \text{Syn}(\mathcal{A})\}$ in time $O(kn)$ and obtain the DFA $\mathcal{A}' = \langle Q, \Sigma \setminus \Gamma, \delta \rangle$ from \mathcal{A} removing the transitions labeled by letters from Γ . It remains to check that \mathcal{A}' is not synchronizing. The latter checking can be done in time $O(kn^2)$ [3]. We have that $rc(L) = 2$ if and only if $rc(L) \neq 1$ and \mathcal{A}' is not synchronizing. \square

We have constructed for each instance of FINITE AUTOMATA INTERSECTION the corresponding automaton \mathcal{A} over the alphabet $\Delta = \Sigma \cup \{x, y, z\}$ in order to prove that SYN-EQUALITY is a **PSPACE**-complete problem. We will use that automaton again to prove the following theorem.

Theorem 4. *Let $J = \text{Syn}(\mathcal{A})$. $\bigcap_{i=1}^n L[M_i] \neq \emptyset$ if and only if $rc(J) > 3$.*

Proof. Let $\bigcap_{i=1}^n L[M_i] = \emptyset$. In this case we have $\text{Syn}(\mathcal{A}) = I$. As it was mentioned above \mathcal{B} is an MSA for I , so we have the equality $rc(\text{Syn}(\mathcal{A})) = 3$. Let us assume now that $\bigcap_{i=1}^n L[M_i] \neq \emptyset$. Since y is a unique reset letter for \mathcal{A} and the automaton \mathcal{A}' (which is obtained from \mathcal{A} by removing of all transitions labeled by y) is still synchronizing, we get that $rc(J) \geq 3$. Now we verify that $rc(J) \neq 3$, that is no 3-state synchronizing automaton is synchronized exactly by J . It is easy to see that if $\bigcap_{i=1}^n L[M_i] \neq \emptyset$, then $J = I \cup I_3 \cup I_4$ where

$$I_3 = \{wz \mid w \in \bigcap_{i=1}^n L[M_i] \text{ and } \delta_i(Q_i, w) \subseteq F_i\};$$

$$I_4 = \{uxwz \mid u \in (\Sigma \cup \{x\})^*, w \in \bigcap_{i=1}^n L[M_i]\}.$$

Arguing by contradiction, assume that $\text{Syn}(\mathcal{B}) = J$ for some 3-state automaton \mathcal{B} over Δ . Denote the state set of \mathcal{B} by $P = \{0, 1, 2\}$ and the transition function by τ . Since y is a reset letter for \mathcal{B} , we have that y brings P to a singleton, say $\{2\}$. Letter z is not reset for \mathcal{A} and $z^2 \in J$, hence z maps P to 2-element subset. It is easy to check that there are just three possible different ways of defining the action of z on the state set P (see Fig.4).

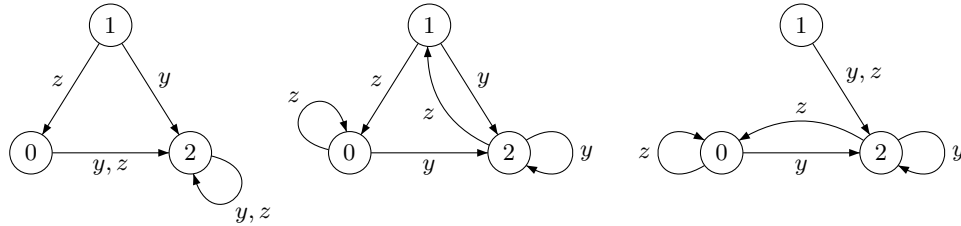


Fig. 4. Possible ways of defining the action of y and z in \mathcal{B} .

Let us assume as above that $\Sigma = \{a, b\}$. It remains to define the action of x, a and b on the state set P . One may see that the words za, zb and zx are reset for \mathcal{A} . So letters x, a and b should map the set $\tau(P, z)$ to singletons. However, any word from $(\Sigma \cup \{x\})^*$ is not reset for \mathcal{A} . In particular, $xx, aa, bb \notin \text{Syn}(\mathcal{A})$. Consider, for instance, the first automaton in the Fig.4. We have that $\tau(P, z) = \{0, 2\}$, thus the action of x, a and b is defined in such way that $|\tau(\{0, 2\}, x)| = |\tau(\{0, 2\}, a)| = |\tau(\{0, 2\}, b)| = 1$. So there are six possible ways of defining the action of x on the states of \mathcal{B} . Since $xx \notin \text{Syn}(\mathcal{A})$, we have that the following two ways of defining the transitions under the action of x are impossible:

012
 x 020
 x 202.

Indeed, in both cases the word xx brings the set $\{0, 1, 2\}$ to a singleton. So, actually, there are just four possible ways of defining the action of x on the states of \mathcal{B} . The same arguments can be provided for letters a and b . Thus the definition of the action of x, a , and b is chosen in one of the following ways:

012
 x_1 010
 x_2 212
 x_3 101
 x_4 121.

For instance, one may say that x acts on P as x_1 , a acts as x_2 and b acts as x_3 . It is sufficient to consider only those cases where all the letters x , a and b act on P differently. There remains four ways of choosing a triple $\{x_i, x_j, x_k\}$ defining the action of letters x , a and b . It can be easily checked that $J \neq \text{Syn}(\mathcal{B})$ in each case. Analogous arguments are provided for the rest two automata in the Fig.4. It means that the reset complexity of the language of reset words of the DFA \mathcal{A} is at least 4, that is $rc(J) \geq 4$. \square

Corollary 2. *Let L be an ideal language and \mathcal{A} a synchronizing DFA over at least 5-letter alphabet with $\text{Syn}(\mathcal{A}) = L$. The problem of checking the inequality $rc(L) \leq 3$ is **PSPACE**-complete.*

Acknowledgment. The author is grateful to participants of the seminar “Theoretical Computer Science” for valuable comments.

References

1. D.S. Ananichev, V.V. Gusev, M.V. Volkov *Slowly Synchronizing Automata and Digraphs*. In: Proc of MFCS 2010. LNCS 6281(010). P. 55–65.
2. J. Černý. *Poznámka k homogénnym experimentom s konečnými automatami*. Mat.-Fyz. Cas. Slovensk. Akad. Vied. 1964 V.14. P. 208–216.
3. D. Eppstein. *Reset sequences for monotonic automata.*, SIAM J. Comput. 1990. V.19. P. 500–510.
4. V. Gusev, M. Maslennikova, E. Pribavkina. *Finitely generated ideal languages and synchronizing automata*. In: J. Karhumäki, L. Zamboni (eds.) Proc. WORDS 2013. LNCS 8079. P. 143–154.
5. D. Kozen *Lower bounds for natural proof systems*. In: Proc. of the 18th FOCS. 1977 P. 254–266.
6. P. Martygin *Computational Complexity of Certain Problems Related to Carefully Synchronizing Words for Partial Automata and Directing Words for Nondeterministic Automata*. In: F. Ablayev (eds.) Theory Comput. Sci. 2014 V.54. P. 293–304.
7. M.I. Maslennikova *Reset Complexity of Ideal Languages*. 2014. arXiv: 1404.2816 (published in M. Bieliková (eds.) Int. Conf. SOFSEM 2012, Proc. V. II, Institute of Computer Science Academy of Sciences of the Czech Republic. 2012 P. 33–44.)
8. J.-E. Pin *On two combinatorial problems arising from automata theory*. Ann. Discrete Math. 1983. V.17. P. 535–548.
9. E.V. Pribavkina, E. Rodaro *Recognizing synchronizing automata with finitely many minimal synchronizing words is PSPACE-complete*. In: B.Löwe (eds.) Proc. CiE 2011. Lect. Notes Comp. Sci., Springer-Verlag, Berlin Heidelberg. 2011 V.6735, P. 230–238.
10. E. Pribavkina, E. Rodaro. *Synchronizing automata with finitely many minimal synchronizing words*. Inf. and Comput. 2011. V.209(3). P. 568–579.
11. S. Sandberg. *Homing and synchronizing sequences.*// In: M. Broy et al (eds.) Model-Based Testing of Reactive Systems, Lect. Notes Comput. Sci, Springer-Verlag, Berlin-Heidelberg-New York. 2005. V.3472. P.5–33.
12. W.J. Savitch. *Relationships between nondeterministic and deterministic tape classes*. J.CSS. 1970. V.4. P. 177–192.
13. M. V. Volkov. *Synchronizing automata and the Černý conjecture*. In: C. Martín-Vide, F. Otto, H. Fernau (eds.), Languages and Automata: Theory and Applications. LATA 2008. Lect. Notes Comp. Sci., Berlin, Springer. 2008. V.5196. P.11–27.